Web aplikacija za upravljanje bazom MP3 pjesama

Petković, Petar

Undergraduate thesis / Završni rad

2016

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:284618

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-05-07**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni preddiplomski studij računarstva

WEB APLIKACIJA ZA UPRAVLJANJE BAZOM MP3 PJESAMA

Završni rad

Petar Petković

Osijek, 2016.

Sadržaj

1. UVOD 1

1.1 Kratak opis zadatka završnog rada 1

2. PROGRAMSKE TEHNOLOGIJE KORIŠTENE U RAZVOJU WEB APLIKACIJE ZA UPRAVLJANJE MP3 PJESAMA 2

2.1 ASP.net MVC i Angular.js MVC 2

2.2 Microsoft SQL server 4

2.3 Razvojno okruženje Microsoft Visual Studio Community 2015 5

3. WEB APLIKACIJA 6

3.1 Pokretanje web aplikacije i postavke servera i baze 6

3.2 Kontrola računa korisnika 10

3.3 Prikazivanje popisa pjesama 13

3.4 Prikazivanje pjesama 16

4. TESTIRANJE RADA I IZGLED WEB APLIKACIJE 19

4.1 Testiranje rada s korisnicima 19

4.2 Testiranje popisa pjesama 20

4.3 Testiranje pjesama 20

4.4 Izgled aplikacije 22

5. ZAKLJUČAK 244

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek,20.09.2016.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada

Ime i prezime studenta:	Petar Petković
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3601, 25.09.2013.
OIB studenta:	42150722197
Mentor:	Doc.dr.sc. Krešimir Nenadić
Sumentor:	
Naslov završnog rada:	Web aplikacija za upravljanje bazom MP3 pjesama
Znanstvena grana rada:	**Informacijski sustavi (zn. polje računarstvo)**
Predložena ocjena završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 Postignuti rezultati u odnosu na složenost zadatka: 3 Jasnoća pismenog izražavanja: 3 Razina samostalnosti: 2
Datum prijedloga ocjene mentora:	20.09.2016.
Datum potvrde ocjene Odbora:	28.09.2016.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTIRADA

Osijek,28.09.2016.

Ime i prezime studenta:	Petar Petković
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3601, 25.09.2013.
Ephorus podudaranje [%]:	4%

Ovom izjavom izjavljujem da je rad pod nazivom**: Web aplikacija za upravljanje bazom MP3 pjesama**

izrađen pod vodstvom mentora Doc.dr.sc. Krešimir Nenadić

i sumentora

mojvlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osimonih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada,osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, ašto je izričito navedeno u radu.

Potpis studenta:

1. UVOD

Web aplikacija za upravljanje mp3 pjesama se sastoji od baze podataka koja sadržava podatke o pjesmama i korisnicima aplikacije. Ta baza će se nalaziti na poslužitelju (engl. *server*) na kojem je programirana web aplikacija u ASP.net tehnologiji programskim jezikom C# (*c-sharp*). Korisnik će bazi podataka pristupati preko web aplikacije. Pristupni dio (engl. *frontend*) web aplikacije napisan u strukturiranom radnom okviru (engl. *frameworku*),skriptnog programskog jezika *Javascript,Angular.js*-u, slat će*ajax* zahtjeve poslužitelju POST metodom i prikazivati odgovore poslužitelja korisniku HTML stranicama. Zbog potrebe za korisnicima bit će omogućeni obrasci za prijavu (engl. *Log In*) i registraciju (engl. *register*). Osim što će korisnici moći pristupati svim svojim pjesmama u bazi podataka, korisnik će moći te pjesme rASPorediti u popise pjesama (engl. *Playlists*) i time lakše naći potrebnu pjesmu. Na sve pjesme koje korisnik spremi u bazu podataka korisnik treba imati autorska prava, tj. korisnik prethodno treba kupiti u ovlaštenim trgovinama pjesmu u mp3 formatu.

1.1 Kratak opis zadatka završnog rada

Izraditi model baze podataka koja će sadržavati popis pjesama u MP3 formatu. Uz naziv pjesme u bazu podataka pohraniti i ostale važne informacije o pjesmi. Izraditi web aplikaciju koja će različitim registriranim korisnicima omogućiti vođenje evidencije svojih MP3 pjesama. Testirati rad web aplikacije i baze podataka.

2. PROGRAMSKE TEHNOLOGIJE KORIŠTENE U RAZVOJU WEB APLIKACIJE ZA UPRAVLJANJE MP3 PJESAMA

U izradi programskog rješenja web aplikacije korišten je radni okvir (engl. *framework*) *Angular.js* koji pomaže u stvaranju dinamičkih web-stranica koje zahtijevaju samo HTML (engl. *HyperText Markup Language*), *CSS* (engl. *Cascading Style Sheets*) i *JavaScript* na klijentskoj strani. Radni okvir je oblik predloška za stvaranje aplikacija koji u sebi uključuje niz programa i skripti kako bi programeru olakšali posao. Pomoću aplikacijskih programskih sučelja (engl. *Application Programming Interface*, skraćeno *API*) su dohvaćene potrebne informacije za rad aplikacije te su potom spremljene u relacijsku bazu podataka *Microsoft SQL*. Za komuniciranje s tom bazom podataka korišten je skriptni jezik ASP.Net te radni okvir uključen unutar *ASP.NET* skriptnog jezika *MVC* (engl. *Model View Controller*). Cijeli kod je napisan u programskom okruženju *Microsoft Visual Studio Community 2015*, a web aplikacija je testirana pomoću *IIS Express* (engl. *Internet Information Services Express*) programa koji stvara lokalni poslužitelj.

2.1 ASP.Net MVC 5 i Angular.js MVC

ASP.NET je razvojna platforma za razvijanje web aplikacija (dinamičkih web stranica). Prva inačica *ASP.NET* 1.0 je izašla u siječnju 2002. godine zamjenjujući klasični *ASP* kao *Microsoftov* poslužiteljski skriptni jezik. Trenutna stabilna inačica je *ASP.NET* 4.5. prema [1]. *ASP.NET* je poslužiteljski jezik što znači da *ASP.NET* ima pristup bazi podataka i datotečnom sustavu web aplikacije na poslužitelju za razliku od skriptnih jezika web preglednika kao što je *Javascript*. *ASP.NET je* popularan zbog razvijenih radnih okvira koji omogućuju lakše razvijanje web aplikacija zbog novih funkcionalnosti koje osnovni *ASP.NET* ne nudi. Za ovaj rad bitan je radni okvir *Entity Framework*zbog kojeg web aplikacija može koristiti *MVC* arhitekturu.

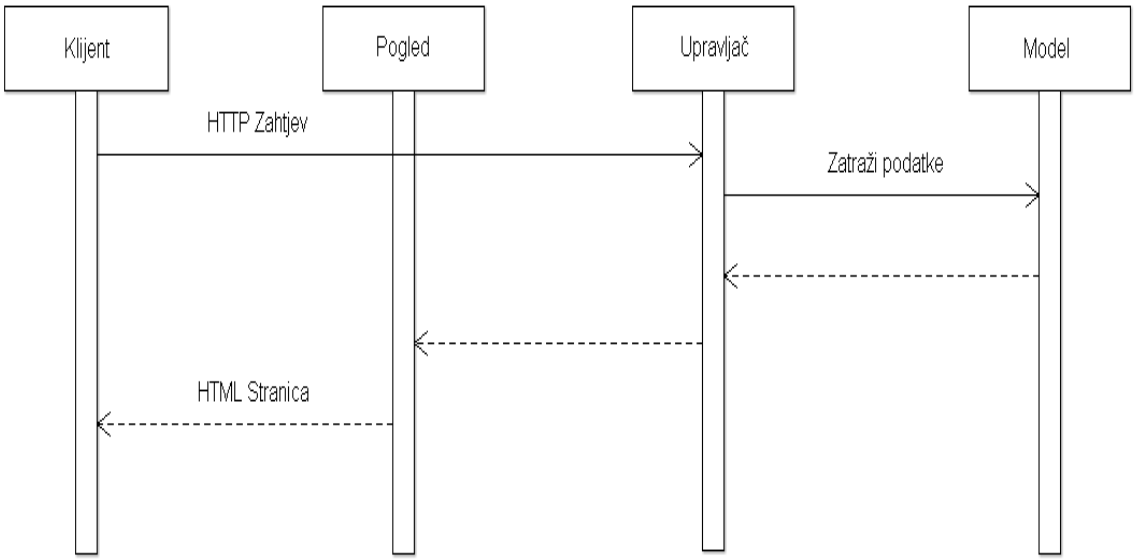
MVC (*Model-View-Controller*) je obrazac programske arhitekture široko prihvaćen za razvoj web aplikacija. Prednost *MVC*-a je što odvaja bilo kakav prikaz informacija od korisničke interakcije s istom, te omogućava nezavisan razvoj, testiranje i održavanje aplikacije. Sastavni dijelovi *MVC*-a su prema [2]:

- Model koji objedinjuje poslovnu logiku i sloj pristupa podacima. U aplikaciji postoji jedan model koji je povezan s bazom podataka, struktura modela je ista kao struktura baze podataka čime je moguće pristupiti entitetima baze podataka preko klasa u modelu *MVC* web aplikacije.

- Pogledi definiraju izgled korisničkog sučelja, prikaz informacija, odnosno traženu web stranicu. Pogledi su *HTML* dokumenti koji sadrže skripte. U aplikaciji su skripte pisane u C# s *RAZORview engine*-om. U web aplikaciji će biti samo dva pogleda u ovom obliku, jedan za registraciju korisnika drugi za prijavu korisnika. Ostali pogledi bit će napisani običnim *HTML* dokumentima a pozivat će ih *AngularJs* direktiva (objašnjeno kasnije u tekstu). Svi pogledi se prikazuju unutar zajedničkog predloška (engl. *Layout*) kako bi aplikacija imala homogeni izgled kroz sve stranice aplikacije.
- Upravljač (engl. *Controller*) sadrži prezentacijsku logiku, obrađuje sve unose korisnika, obavlja izmjenu podataka s modelom i određuje pogled. Upravljač sadrži više akcija koje se izvode ovisno o unosu korisnika. Uz to što određuje pogled upravljač može vratiti i *JSON* (engl. *Javascript Object Notation*) što je format koji šalje ljudima čitljiv tekst u obliku parova atribut-vrijednost prema [3]. Unutar rada samo će *AccountController* (poglavlje 3) slati poglede dok će ostali upravljači slati ili JSON ili će preusmjeriti na neki drugi upravljač.

MVC arhitektura se koristi i u Angular.js s tom razlikom da Angular.js ne može izravno pristupiti bazi pa se zbog toga mora služiti *HTTP GET* i *POST* metodama koji šalju zahtjev *ASP.NET* upravljaču od kojeg onda dobiju potrebne podatke u *JSON* obliku i prezentira ih pomoću HTML dokumenata (pogleda) prema [4]. U ovom radu se uz *ASP.NET* MVC koristi i *Angular.js MVC* zbog toga što će se time slati manje zahtjeva klijentske strane serveru i samo onda kad je to potrebno, uz to će se osvježiti samo *HTML* element koji je obavio zahtjev a ostali elementi će ostati netaknuti prema [5].

Osim upravljača AccountController (poglavlje 3) svi će *ASP.NET* upravljači imati svoj upravljač u Angular.js aplikaciji na klijentskoj strani. Način na koji se pristupa stranicama unutar *ASP.NETMVC* aplikacije vidljiv je na slici 2.1.



Sl. 2.1 Dohvat stranice u MVC arhitekturi

Prema slici 2.1 vidi se da klijent traži podatke od upravljača, koji ih dobavlja iz modela, nakon toga upravljač šalje pogled s*HTML* stranicom. Najveća mana *MVC* arhitekture je puno zahtjeva klijenta poslužitelju zbog čega su mogući zastoji na poslužitelju.

2.2 Microsoft SQL server

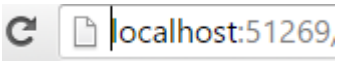
Baza podataka iz koje će se razviti model logike aplikacije je napravljena u *Microsoft SQL* serveru integriranom u programskom okruženju *Microsoft Visual Studio 2015*. *Microsoft SQL* server se služi relacijskim bazama podataka kojima je s *IIS Express* serverom povezan s aplikacijom prema[6]. *IIS server* zna s kojom je bazom podataka povezana web aplikacija preko *Connection Stringa*, što su linije koda u *Web Config* datoteci ASP.Net MVC projekta kojom se povezuju baza podataka i *MVC* aplikacija prema[7]. *Connection String*kao parametre prima ime baze podataka, lokaciju baze i zaporku i korisničko ime ako je to potrebno za pristup bazi. Baza podataka korištena u Web aplikaciji je „BazaMp3Entities“ a *Connection string* za bazu u *Web Configu* vidljiv je na slici 2.2.

```
<connectionStrings>
  <add name="BazaMp3Entities"
   connectionString="metadata=res://*/Models.Mp3Model.csdl|res://*/Models.Mp3Model.ssdl|res://*/Models.Mp3Model.msl;
   connection string=&amp;quot;data
   source=(LocalDB)\MSSQLLocalDB;attachdbfilename=|DataDirectory|\BazaMp3.mdf;integrated
   security=True;MultipleActiveResultSets=True;App=EntityFramework&amp;quot;"
  providerName="System.Data.EntityClient" />
</connectionStrings>
```

Sl. 2.2 Connection string baze podataka

2.3 Razvojno okruženje *Microsoft Visual Studio Community 2015*

Cijeli kod napisan u ovom radu napisan je *Microsoft Visual Studio Community edition* 2015 razvojnom programskom okruženju. Razvojno okruženje *Microsoft Visual Studio 2015* u suradnji s programima *Microsoft SQL* server 2012 i *IIS Express* serverom omogućuje pisanje i testiranje svog potrebnog koda za ovu aplikaciju prema [8]. Testiranje se vrši tako da *Visual Studio* s *IIS expressom* napravi lokalni web poslužitelja na određenom portu kojem se pristupa u web pregledniku kao na slici 2.3:



Slika 2.3 Port lokalnog poslužitelja

Nakon adrese lokalnog poslužitelja upisivat će se upravljač pa akcija unutar upravljača.

3. WEB APLIKACIJA

Proces izrade web aplikacije podijeljen je u 4 poglavlja. Svako poglavlje će sadržavati određeni logički dio web aplikacije. Prvi dio će objašnjavati pokretanje web aplikacije i postavke poslužitelja i baze podataka. U drugom dijelu će se objasniti kontrolu računa korisnika. U trećem dijelu kontrola popisa pjesama. U četvrtom dijelu kontrola pjesama unutar popisa pjesama.

3.1Pokretanje web aplikacije i postavke servera i baze

Pri pokretanju web aplikacije u web serveru je namješteno da se prvo pokrene datoteka Global.asax.cs u pozadini, koja u sebi sadržava instrukcije za pokretanje koda sa postavkama bitnih datoteka u projektu. Tipovi datoteka koje pokrene Global datoteka su: postavke ruta, postavke paketa i postavke filtera. Postavke filtera za ovu aplikaciju nisu toliko bitne, ali zbog korištenja *Angular.js* klijentske strane su jako bitne postavke paketa. Paket je zapravo popis svih bilbioteka skripti koji će se koristiti unutar aplikacije, od tih skripti se prave paketi koji se kasnije mogu jednostavno pozvati unutar *HTML* pogleda. Učitavanje svih paketa korištenih u programu je vidljivo na slici 3.1.

```
bundles.Add(new ScriptBundle("~/bundles/angularjs").Include(
            "~/Scripts/angular.js",
            "~/Scripts/angular-cookies.js"));

bundles.Add(new StyleBundle("~/Content/user/css").Include(
          "~/Content/CustomStyles/*.css"));

bundles.Add(new ScriptBundle("~/bundles/appscripts").Include(
            "~/AngularApp/*.js",
            "~/AngularApp/Controller/*.js",
            "~/AngularApp/Directive/*.js",
            "~/AngularApp/Factory/*.js"
            ));
```

Sl. 3.1 Učitavanje paketa Angular.js aplikacije

Ovakvim pisanjem koda, u pakete se učitavaju sve datoteke s nastavkom - .js unutar svih navedenih mapa. Zbog toga nije potrebno nadodati datoteku svaki put pojedinačno u pakete već će se to automatski napraviti.

Postavke ruta su prilagođene načinu na koji MVC arhitektura radi a kod postavki je prikazan na slici 3.2.

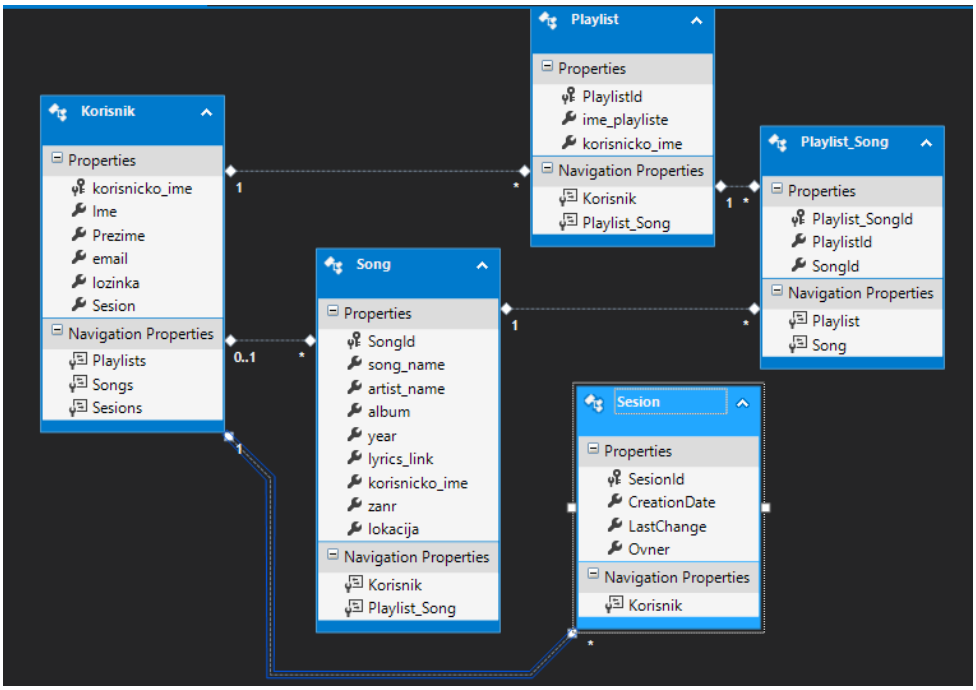
```
routes.MapRoute(
    name: "Default",
    url: "{controller}/{action}/{id}",
    defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }
);
```

Sl. 3.2 Postavke ruta

Unutar koda se definira način na koji se pozivaju pojedine akcije upravljača i način na koji se šalju parametri pojedinoj akciji. Definiran je i upravljač koji se poziva kad se pozove lokalni poslužitelj.

Nakon osnovnih postavki poslužitelja i globalnih datoteka, uspostavlja se veza baze s aplikacijom (postupak je vidljiv u poglavlju 2). Baza podataka je sastavljena od 5 entiteta (tablica). Web aplikacija je za upravljanje baze Mp3 pjesama zbog čega je napravljena tablica *„Song“* koja sadržava osnovne informacije o svakoj pjesmi i mjesto gdje se ta pjesma nalazi na poslužitelju. Svaki član tablice *„Song“* ima svog vlasnika iz tablice *„Korisnik“* koja sadržava osnovne informacije o korisnicima web aplikacije. Svaki korisnik također može biti vlasnik članova u tablici *„Playlist“* koja sadržava popise pjesama. Zbog praćenja povezivanja korisnika na bazu podataka napravljena je tablica *„Sesion“* koja u sebi sadrži sva spajanja korisnika koja se bilježe zbog dodavanja mogućnosti korisniku da se ne odjavi sa stranice i da ostane prijavljen uvijek, a da se zadrži sigurnost i integritet podataka. U tablici *„Playlist-Song“* su povezane stavke tablice *„Playlist“* i *„Song“*jer između te dvije tablice postoji veza više na više zbog koje je potrebno praviti novi entitet tj. tablicu.

UML model baze podataka prikazan je na slici 3.3.



Sl. 3.3 UML model baze podataka

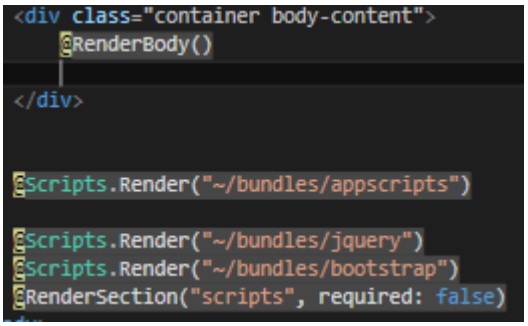
Da bi se ta baza mogla jednostavno koristiti u kodu, od nje se pravi *ADO.NETentity data model* tj. prave se klase od tablica u bazi podataka. Stvorene klase se kasnije koriste pri ubacivanju, brisanju i prikazivanju podataka u bazi. *ADO.NET entity dana model* je funkcija ugrađena u *Visual Studio 2015* (prema [9]) a da bi se mogao koristiti unutar projekta potrebno je uključiti *Entity Framework* biblioteku koju je potrebno skinuti s *Microsoftove* web stranice prema [10]. Izgled klasa dobivenih *ADO.NETentity dana model* može se vidjeti na slici 3.4.

```
public partial class BazaMp3Entities : DbContext
{
    public BazaMp3Entities()
        : base("name=BazaMp3Entities")
    {
    }

    protected override void OnModelCreating(DbModelBuilder modelBuilder)
    {
        throw new UnintentionalCodeFirstException();
    }

    public virtual DbSet<Korisnik> Korisniks { get; set; }
    public virtual DbSet<Playlist> Playlists { get; set; }
    public virtual DbSet<Playlist_Song> Playlist_Song { get; set; }
    public virtual DbSet<Song> Songs { get; set; }
    public virtual DbSet<Sesion> Sesions { get; set; }
}
```

Sl. 3.4 Klasa BazaMp3Entities

Najbitnija klasa stvorena *ADO.NET entity modelom* je *BazaMp3Entities* kojom je moguće unutar koda obaviti bilo koji upit bazi podataka jer ova klasa predstavlja i vezu s Bazom *„Mp3DatabaseEntities“*.Unutar svakog upravljača će se kreirati objekt klase DatabaseMp3Entities kojim će upravljač imati vezu s modelom podataka.

Poslije modela podataka koji će se koristiti u svim dijelovima *ASP.NET* web aplikacije potrebno je napraviti *Layout* stranicu, to je *HTML* stranica koja ima elemente što se nalaze unutar svih pogleda (kao na slici 3.5), ti elementi su zaglavlje i podnožje stranice. *Layout* stranica je dobro mjesto za prizivanje paketa napravljenih u *BundleConfig.cs* datoteci c# funkcijom nutar *HTML*-a *Scripts.Render*, u *Layout* stranici *MVC* aplikacije mora postojati i naredba *RenderBody* koja na mjesto na kojem je postavljena učitava pogled koji je poslao upravljač.U *RAZOR HTML* dokumentu se C# dio koda naznači s @ znakom.



Sl. 3.5 Layout HTML stranica

Na početku aplikacije još je potrebo i definirati Angular.js aplikaciju te ju pozvati unutar HTML pogleda na poslužitelju. Ime Angular.js aplikacije bit će *„MainApp“* kao na slici 3.6.

```
var app = angular.module('MainApp', ['ngCookies']);
```

Sl. 3.6 Definicija Angular.js aplikacije

Pri definiranju aplikacije potrebno je navesti u uglatim zagradama o kojim će bibliotekama Angular.js-a aplikacija ovisiti, ova aplikacija će kasnije koristiti *HTTP* kolačiće. *HTTP* kolačić (engl. *cookie*) je tekstualna datoteka u koju se može spremiti neka vrijednost na web pregledniku, u ovom radu će se u *HTTP* kolačić spremati vrijednost sjednice čime će se provjeravati je li korisnik prijavljen. Postoji biblioteka koja se nosi s pristupom *HTTP* kolačićima u *Angular*.js, naziva se *„ngCookies“*pa se stavljau ovisnosti aplikacije MainApp. Nakon definiranja aplikacije, aplikacija se poziva unutar *HTML* datoteke kao na slici 3.7.

```
<div ng-app="MainApp">
    <organizer></organizer>
</div>
```

Sl. 3.7 Poziv Angular.js aplikacije

Elementima kasnije kreiranim unutar MainApp aplikacije moguće je pristupiti samo unutar tog bloka. Na početku je odmah stvorena i prva Angular.js direktiva *„orginizer“*. Direktiva (engl. *directive*) je proširenje osnovne sintakse *HTML*-a, kojim se *HTML* elemente „uči“ nove načine ponašanja, a moguće je stvoriti i definirati nove *HTML* elementa što je vidljivo u kodu na slici 3.7 prema [11]. Direktiva orginizer je postavljena kao novi element u *HTML*-u a unutar predloška direktive bit će pozvane druge direktive, direktive za pjesme i direktive za popise pjesama kao na slici 3.8.

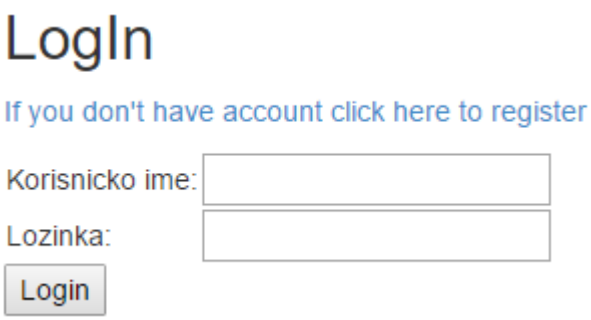
```
<div class="Organizer">
    <table>
        <tr>
            <td>
                <playlist  selected="var_selectedPlaylist" createdsong="var_createdSong"></playlist>
            </td>

            <td>
                <songlist  playlist="var_selectedPlaylist" selected="var_selectedSong" created="var_createdSong" ></songlist>
            </td>
        </tr>
    </table>
</div>
```

Sl.3.8 Predložak direktive orginizer

Unutar poziva svake direktive (HTML elementa) mogu se proslijediti varijable drugim direktivama. Te varijable se proslijeđuju uglavnom zbog toga da postanu okidač budućih događaja u upravljačima drugih direktiva. Svaka direktiva Angular.js-a ima i svoj upravljač, s obzirom da se unutar orginizer direktive ne prikazuje sadržaj nikakvog modela ništa značajno nije napravljeno unutar upravljača orginizer direktive.

3.2Kontrola računa korisnika

Pri ulasku na početnu stranicu web aplikacije pretražuje se kolačić u kojem se nalazi informacija o tome jeli neki račun trenutno prijavljen na web aplikaciju u web pregledniku. Ukoliko postoji prijavljen račun web aplikacija usmjerava korisnika do popisa njegovih pjesama te mu omogućava uređivanje, brisanje i dodavanje novih pjesama i popisa pjesama. Ukoliko ne postoji prijavljen račun korisnik se usmjerava na pogled za prijavu (engl. *Log In*) kao na slici 3.9.



Sl. 3.9 Pogled LogIn

U pogledu *LogIn* se nudi mogućnost registracije ako korisnik još uvijek nema račun. Ako korisnik ima račun treba upisati svoje korisničko ime i zaporku da bi se prijavio, a nakon toga kliknuti tipku *LogIn.*Nakon što se klikne tipka *LogIn* pogled šalje *HTTP* ajax POST metodom zahtjev metodi *LogIn* unutar *Account* upravljača, s parametrima koje je korisnik ispunio.U *LogIn* metodi se prvo provjerava jesu li sva polja popunjena, ukoliko neko polje nije popunjeno šalje se korisniku isti pogled s porukom pogreške. Ukoliko su sva polja popunjena unesene vrijednosti

uspoređuju se s onima u bazi podataka. Da bi se vrijednosti usporedile s onima u bazi podataka prvo je potrebno dohvatiti vrijednosti iz baze što se radi upitom kao na slici 3.10.

```
string query = @"SELECT Korisnik.lozinka
                 FROM Korisnik WHERE Korisnik.korisnicko_ime = @korisnik";
SqlParameter[] param = new SqlParameter[1];
param[0] = new SqlParameter("@korisnik", log.UserName);
IEnumerable<string> res = db.Database.SqlQuery<string>(query, param);
```

Sl. 3.10 MSSQL upit

U kodu na slici 3.10 db je objekt klase BazaMp3Entities tj. modela baze podataka. Za postavljanje upita se poziva metoda tog objekta *Database.SqlQuery*, ta metoda služi za postavljanje upita samoj bazi podataka a kao parametre prima tekst upita i parametre potrebne za upit. Tekst upita se nalazi u varijabli tipa string query, a parametri se definiraju kao liste *SqlParameter* objekata koji koriste *System.Data.SqlClient* biblioteku.Rezultat metode *Database.SqlQuery* je lista tipa zadanog između znakova manje i više.Svi upiti i dalje u radu se prave na ovakav način, iznimka će biti naredbe bazi podataka tj. unošenje i brisanje stavki u i iz baze podataka, za što će se koristiti *Database.SqlCommand* metoda, koja kao odgovor daje logičku nulu ili jedinicu u ovisnosti izvršavanja naredbe. Cilj upita na slici 3.10. je naći zaporku iz baze podataka za zadano korisničko ime, ukoliko postoji zaporka za uneseno korisničko ime provjerava se ispravnost zaporke. Zbog sigurnosnih razloga zaporka je u bazu podataka spremljena kao rezultat hash funkcije. Hash funkcija kriptira zaporku u niz nasumičnih znakova s točno određenim brojem znakova (u ovom slučaju 100), time se u slučaju provale u bazu podataka štiti zaporku svakog korisnika i ukoliko se ne koristi ista hash funkcija kao u aplikaciji nemoguće je naći zaporku. Hash zaporke dobiven iz baze podataka se uspoređuje sa stvarnom zaporkom funkcijom CheckHash koja od unesene zaporke pravi hash i ukoliko taj hash bude jednak hashu iz baze zaporka je potvrđena i ide se u daljnji postupak prijave. U daljnjem postupku prijave se u tablicu Sesija ubacuje nova sjednica, a u kolačić web preglednika se upisuje identifikacijska oznaka sjedniceupravo umetnute u tablicu sjednica (kao na slici 3.11).

```
log.Error = "bez greske";
string command = @"INSERT INTO Sesion(SesionID,CreationDate,LastChange,Ovner)
                   VALUES(@ID,@sada,@sada,@ime)";
SesionID id = new SesionID();
string hash = id.Hash();
SqlParameter[] cparam = new SqlParameter[3];
cparam[0] = new SqlParameter("@ID",hash);
cparam[1] = new SqlParameter("@sada", System.DateTime.Now);
cparam[2] = new SqlParameter("@ime", log.UserName);
db.Database.ExecuteSqlCommand(command, cparam);
HttpCookie cookie = new HttpCookie("UserName");
cookie.Value = hash;
cookie.Expires.AddYears(50);
Response.Cookies.Add(cookie);
command = @"SELECT Korisnik.Ime AS Name, Korisnik.Prezime AS LastName
            FROM Korisnik WHERE Korisnik.korisnicko_ime=@ime";
param[0] = new SqlParameter("@ime", log.UserName);
IEnumerable<HomeIndexModel> odg = db.Database.SqlQuery<HomeIndexModel>(command, param);
HomeIndexModel wc = new HomeIndexModel();
foreach(HomeIndexModel nl in odg) { wc = nl; }
return RedirectToAction("Index", "Home",wc);
```

Sl. 3.11 Krajnja prijava korisnika

Poslije upisivanja u kolačić korisnik se preusmjerava na početnu stranicu gdje se učitava *Angular.js* aplikacija. U slučaju da korisnik posjeti web aplikaciju dok je još prijavljen Home upravljač će provjeriti jeli korisnik prijavljen tj. ima li nešto zapisano u kolačiću (slika 3.12).

```
if (Request.Cookies["UserName"] != null)
{

    {
        return RedirectToAction("CheckHashLogIn", "Account");
    }
}
return RedirectToAction("Login", "Account", null);
```

Sl3.12 Provjera kolačića

Ako Home upravljač (metoda indeks) nađe nešto zapisano u kolačiću usmjerit će program prema metodi *CheckHashLogIn*, ta metoda provjerava ispravnost kolačića, i prema nazivu unutar kolačića traži korisničko ime kao na slici 3.13.

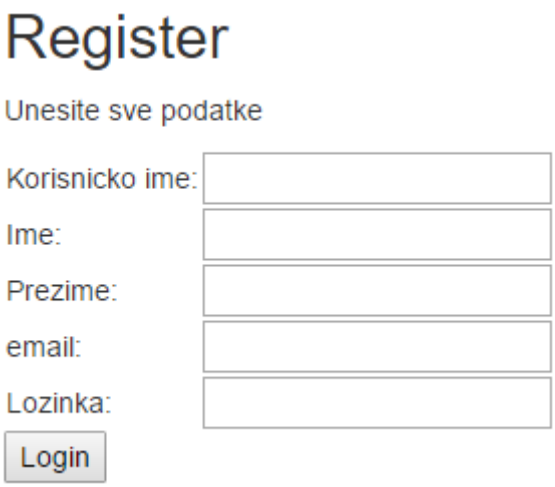
```
string command = @"SELECT Sesion.Ovner FROM Sesion
                   WHERE Sesion.SesionId=@ID";
SqlParameter[] param = new SqlParameter[1];
param[0] = new SqlParameter("@ID", UserName);
IEnumerable<string> names = db.Database.SqlQuery<string>(command, param);
List<string> namez = names.ToList();
```

Sl. 3.12 Provjera kolačića

Ukoliko je ime sjednice spremljeno u kolačić ispravno, CheckHashLogIn metoda će unutar upita dobiti jedan rezultat i preusmjeriti tijek programa prema *WebUser* metodi Home upravljača jer se

u pogledu koji vraća ta metoda nalazi *Angular.js* web aplikacija u kojoj korisnik može upravljati svojim pjesmama.

Kad korisnik nije registriran treba pristupiti obrascu za registraciju koji izgleda kao slika 3.13., ondje korisnik treba unijeti svoje podatke, među kojima je i korisničko ime koje mora biti različito od svih dosad unešenih korisničkih imena.



Sl. 3.13 Obrazac za registraciju

Nakon što korisnik popuni sva polja unešeni podaci se unose i u bazu podataka. Uz unošenje podataka u bazu metoda register i napravi mapu u koju će se spremati pjesme korisnika kao na slici 3.14, i napravit će prvi popis pjesama gdje će se povezivati sve pjesme korisnika.Nakon što je stvorena mapa za korisnika i podaci uneseni u bazu poziva se *LogIn* metoda s parametrima novog korisnika.

```
string pathToCreate = "~/Songs/" + reg.UserName;
Directory.CreateDirectory(Server.MapPath(pathToCreate));
```

Sl. 3.14 Kreiranje korisničke mape

Za odjavljivanje korisnik treba kliknuti na hipervezu Odjavi se, a tamo se poziva *LogOut* metoda Account upravljača. U *LogOut* metodi se briše podatak u kolačiću i red iz sesije gdje su informacije o prethodnoj prijavi korisnika.Korisnik se preusmjerava na *LogIn* stranicu.

3.3Prikazivanje popisa pjesama

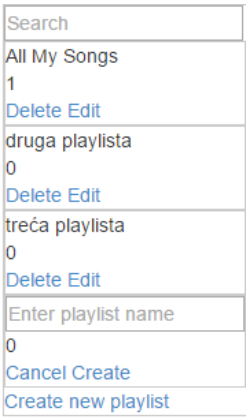
Nakon prijave se otvara Angular.js web aplikacija tj. direktiva orginizer (poglavlje 3.1) se pokreće. Unutar direktive orginizer pozvane su druge dvije direktive, direktiva playlist i direktiva songlist. Direktiva playlist služi za prikazivanje popisa pjesama koji će u web aplikaciji služiti korisniku za bolje snalaženje sa svojim pjesmama. Playlist direktiva čim je pokrenuta

šalje *HTTP* ajax POST zahtjev GetPlaylist metodi playlist upravljača na poslužitelju. *HTTP* zahtjev se definira pod funkciju GetPlaylist u upravljaču playlist direktive kao na slici 3.15. Parametri potrebni *HTTP* POST zahtjevu su adresa na koju se šalje zahtjev i podaci koji se šalju.

```
$scope.GetPlaylists = function () {
    prepData = {
        UserName: UserManager.GetUserId()
    };
    $http({
        url: "/Playlist/GetPlaylists",
        method: "POST",
        data: prepData,
    }).success(function (data) {
        $scope.MyData = data;
    }).error(function (err, status) {
        $scope.error_msg.error = true;
        $scope.error_msg.status = status;
        $scope.error_msg.msg = "Failed to get playlist data";
    })
};
```

Sl. 3.15 GetPlaylist funkcija

HTTP zahtjev nakon što je pokrenut može imati dva ishoda, može biti uspješan i tad će se otvoriti suucces funkcija iz koje će dobiveni podaci biti popisi pjesama prikazani u *html* predlošku playlist direktive pod imenom MyData, ukoliko je došlo do pogreške pri vezi sa serverom sadržaj poruke se ispiše na korisnički ekran što je dio .error funkcije.



Sl. 3.16 Popisi pjesama

Za pristup *html e*lementima ili varijablama unutar direktive popis pjesama koristi se oznaka $scope i nakon nje ime elementa.HTML predložak direktive playliste ima izgled kao slika 3.16 a sastoji se od nekoliko elemenata: popisa pjesama, opcija za dodavanje i ažuriranje imena popisa pjesama te obrasca za unos novog popisa koji se prikazuje samo kad je pritisnut link *Creat new playlist.*

Za prikaz popisa pjesama koriste se podaci dobiveni GetPlaylist metodomi ng-repeat direktiva Angullar.js-a koja nudi pristup svakom objektu unutar liste MyData kao na slici 3.17. Za prikaz

pojedinog elementa popisa pjesama korištena je direktiva playlist-tile, koja osim što prikaže svaki popis nudi i mogućnost uređivanja svakog popisa pjesama. Novoj direktivi trebaju se proslijediti podaci unutar MyData preko ng-repeat direktive da bi playlist-tile direktiva mogla pokazivati podatke dobivene sa servera.

```
<div ng-repeat="item in MyData | filter:var_search">
    <playlist-Tile model="item"></playlist-Tile>
</div>
```

Slika 3.17 ng-repeat direktiva

Uz ng-repeat direktivu korišten je filter koji će filtrirati popis pjesama prema unesenom nazivu u mjestu za tekstualni unos var_search. Create playlist funkcija Angular.js aplikacije se poziva unutar obrasca za stvaranje popisa pjesama na hipervezu Create, pozivom te funkcije se šalje *HTTP* zahtjev CreatePlaylist metodi Playlist upravljača čime se stvara novi popis pjesama. CreatePlaylist, GetPlaylist i UpdatePlaylist metode Playlist upravljača samo sadrže MSSQL upite i naredbe kojima utječu na bazu podataka stoga nisu posebno objašnjavani unutar rada, kod za cijeli upravljač je prikazan uprilozima. Pri odabiru opcije Edit u popisu pjesama omogućava se uređivanje pojedine playliste, prilikom klika na tu vezu u *HTML* stranici se pojavljuje skriveni element koji nudi uređivanje imena popisa pjesama a izgleda kao slika 3.18.

Sl. 3.18 Obrazac za uređivanje

Nadalje će se klikom na Save opciju pozvati UpdatePlaylist metoda i time će se ažurirati podaci tablici.

Na slici 3.18 je vidljivo da se promijenila boja popisa koji se uređuje u odnosu na ostale popise, to se događa zbog stila korištenog za označene elemente u HTML dokumentu. Za označavanje elementa napravljena je posebna tvornička postavka u Angular.js aplikaciji, uz označavanje elemenata u tvorničkim postavkama su objašnjena ponašanja kao što su brisanje elementa, dodavanje novog elementa itd. Uz tvorničku postavku koja govori koji je element označen najbitnija je tvornička postavka koja provjerava kolačić s imenom trenutne sesije koji se šalje u svim *HTTP* POST zahtjevima poslužitelju za provjeru identiteta korisnika čime je osiguran sigurnost i integritet podataka. Podatak o označenom popisu pjesama dostupan je i upravljaču songlist direktive, a prenosi se u html predlošku orginizer direktive kao na slici 3.8.

3.4 Prikazivanje pjesama

Za prikaz pjesama se koristi direktiva songlist. Upravljač direktive prikazuje pjesme u odnosu na označen popis pjesama, ukoliko nijedan popis pjesama nije označen ne prikazuje se ni jedna pjesma. Pri otvaranju songlist direktive odmah se šalje *HTTP* zahtjev za GetSonglist metodu Song upravljača na serveru, kao parametri se šalju identifikacijska oznaka Popisa pjesama i ime sesije iz kolačića. Sam zahtjev izgleda identično kao zahtjev za dobiti popise pjesama pa za postupak se može vidjeti poglavlje 3.3. Za prikaz pjesme je potrebno više parametara nego što je bilo potrebno za prikaz popisa pjesama jer pjesme imaju više parametara u bazi podataka. Izgled html stranice popisa pjesama iz određenog popisa vidljiv je na slici 3.19.

Sl. 3.19 Izgled popisa pjesama

Pomoću filtra ng-repeat direktive je omogućena pretraga pjesama kao što je to bilo moguće u popisima pjesama. Pri stvaranju nove pjesme se otvara obrazac kao na slici 3.20., iz tog obrasca se vidi da za razliku od obrasca za stvaranje popise pjesama svi argumenti nisu tekstualnog oblika jer postoji i element za mp3 datoteku koja će se spremiti na poslužitelj.

Sl. 3.20 Obrazac za dodavanje pjesme

Zbog toga što postoji i datoteka u parametrima za slanje poslužitelju potrebno je napraviti direktivu za model koji može raditi i s datotekama. Kako se priziva taj model i kako se priziva običan model podataka u angular aplikaciji prikazano je na slici 3.21.

```
<div class="Songlist-tile-display">
    <div>
        <input id="SongName" name="SongName" placeholder="Enter song name" type="text" data-ng-model="model.SongName" ng-init="model.SongName=''"/>
    </div>
    <div>
        <input id="SongArtist" name="SongArtist" placeholder="Enter song Artist" type="text" data-ng-model="model.SongArtist" ng-init="model.SongArtist=''"/>
    </div>
    <div>
        <input id="AlbumName" name="AlbumName" placeholder="Enter album name" type="text" data-ng-model="model.AlbumName" ng-init="model.AlbumName=''"/>
    </div>
    <div>
        <input id="AlbumYear" name="AlbumYear" placeholder="Enter album year" type="text" data-ng-model="model.AlbumYear" ng-init="model.AlbumYear=''"/>
    </div>
    <div>
        <input id="SongGenre" name="SongGenre" placeholder="Enter song genre" type="text" data-ng-model="model.SongGenre"  ng-init="model.SongGenre=''"/>
    </div>
    <div>
        <input  type="file" data-ak-file-model="model.attachment"  />
    </div>
```

Sl. 3.21. HTML kod za obrazac stvaranja pjesme

U kodu se obični angular model priziva linijom dana-ng-model=“model.ime_modela“ ali obični angular model ne podržava datoteke zbog čega je potrebno napraviti novu direktivu koja će omogućiti angularu rad s datotekama. Ime te direktive je akfilemodel a u html-u se ona priziva kao na slici 3.21. Akfilemodel direktiva učitava odabranu datoteku i omogućuje toj datoteci slanje na poslužitelj uz ostale elemente modela. Kod za akfilemodel direktivu nalazi se na slici 3.22.

```
.directive("akFileModel", ["$parse",
        function ($parse) {
            return {
                restrict: "A",
                link: function (scope, element, attrs) {
                    var model = $parse(attrs.akFileModel);
                    var modelSetter = model.assign;
                    element.bind("change", function () {
                        scope.$apply(function () {
                            modelSetter(scope, element[0].files[0]);
                        });
                    });
                }
            };
        }]);
```

Sl.3.22 Akfilemodel direktiva

Uz ovu direktivu napravljena je i funkcija kojom se u jedan *JSON* spremaju angular model i novonapravljeni model. Ta funkcija sve tekstualne dijelove modela i datoteke stavlja kao pojedini objekt u formdata varijablu *anguular.js*-a kao na slici 3.23.

```
var getModelAsFormData = function (data) {
    var dataAsFormData = new FormData();
    angular.forEach(data, function (value, key) {
        dataAsFormData.append(key, value);
    });
    return dataAsFormData;
};
```

Sl. 3.23 Dodavanje u formdata

Formdata je oblik angular.js varijable koji je pogodan za slanje *HTTP* metodama prema [11]. Song Upravljač na poslužitelju za rad s datotekom koristi klasu httpPostedFileBase koja omogućuje pristup dijelovima datoteke kao što su ime te spremanje datoteke u korisnikov folder na server.

```
string oldfilepath = "~/Songs/" + owner + '/' +Path.GetFileNameWithoutExtension(a.attachment.FileName);
string newfilepath = oldfilepath+".mp3";
int fileint = 1;
while (System.IO.File.Exists(Server.MapPath(newfilepath))){
    newfilepath = oldfilepath + "(" + fileint.ToString() + ").mp3";
    fileint++;

};

string command = @"INSERT INTO Song(song_name,artist_name,album,year,korisnicko_ime,zanr,lokacija)
                   VALUES(@ime,@art,@alb,@god,@vlas,@zanr,@lok)";
a.attachment.SaveAs(Server.MapPath(newfilepath));
```

Sl. 3.24 Spremanje pjesme na poslužitelj

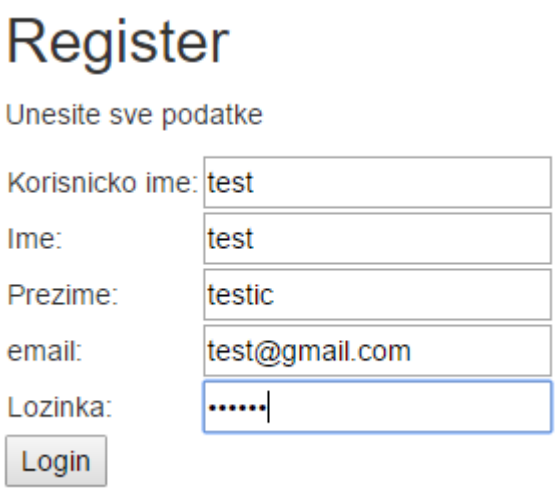
Na slici 3.24 je prikazan kod za spremanje datoteke na poslužitelj u korisnikovu mapu. Za sepremanje se koristi SaveAs metoda objekta httpPostedFileBase klase prije koje je potrebno provjeriti postoji li već datoteka s istim imenom spremljena na poslužitelj. Ukoliko postoji datoteka s takvim imenom u korisničkoj mapi na poslužitelju potrebno je datoteku spremiti s novim imenom. Ruta za novo ime pjesme se sprema u bazu podataka kao lokacija pjesme.

Sve pjesme je moguće dodati u bilo koji popis pjesama, i svaku novu pjesmu je moguće napraviti u bilo kojem popisu pjesama a ukoliko taj popis nije korisnikov sa svim pjesmama pjesma će se tamo automatski dodati sama, ako se pjesma pravi kada nijedan popis pjesama nije označen pjesma se samo sprema u popis sa svim pjesmama.

4. TESTIRANJE RADA I IZGLED WEB APLIKACIJE

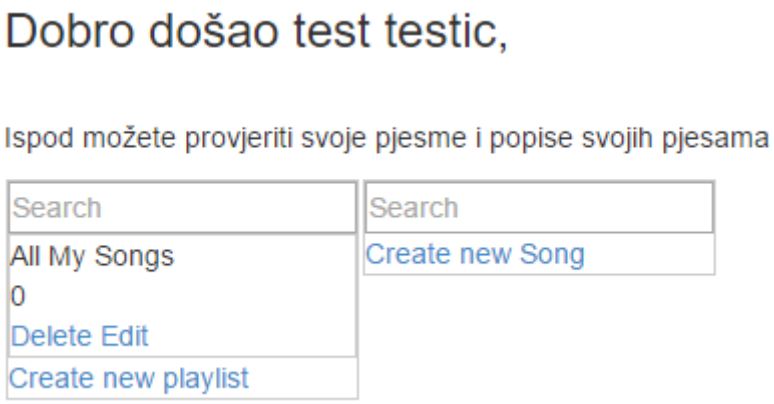
4.1Testiranje rada s korisnicima

Za testiranje rada s korisnicima ispunit će se obrazac registracije i prijaviti novi s novim korisničkim računom. Na slici 4.1 vide se uneseni podaci u html obrazac za registraciju (unesena zaporka je 123456).



Sl. 4.1 Testni obrazac

Klikom na login otvori se nova stranica (slika 4.2.) koja pozdravi korisnika i pokaže stvoren popis pjesama koji se radi pri registraciji, na slici 4.3 se vidi da je novi korisnik unesen u bazu podataka.



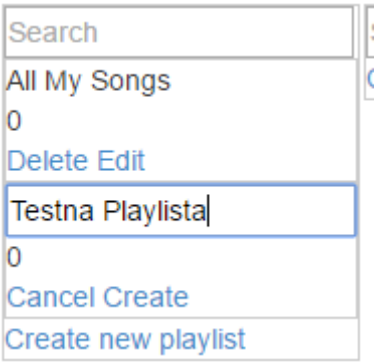
Sl. 4.2 Pozdravna stranica

	korisnicko_ime	Ime	Prezime	email	lozinka	Sesion
	josko	josko	lokas	adjkjsd@mail.c...	1000:FfXPG7q4...	1
	joskovbrat	brat	lokas	adjkjsd@mail.c...	1000:ahiGMwc...	1
	new_user	New	User	a@email.com	1000:Lo1dvfV5a...	42
	PUBLIC	OWNER	EVERYTHING	owner@gmail.c...	1000:SWPyaQW...	2
	test	test	testic	test@gmail.com	1000:DILWafC0...	1

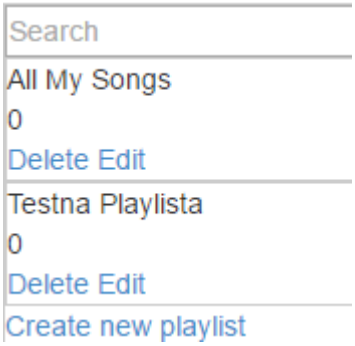
Sl. 4.3 Novi podaci u bazi

4.2Testiranje popisa pjesama

Za testiranje popisa pjesama stvorit će se novi popis pjesama za korisnika test, nakon čega će se pokazati da je taj popis dodan u bazu. Na slici 4.4. vidi se ispunjeni obrazac za novi popis a na slici 4.5. vidi se dodan popis pjesama na samoj stranici.



Sl. 4.4. Nova playlista



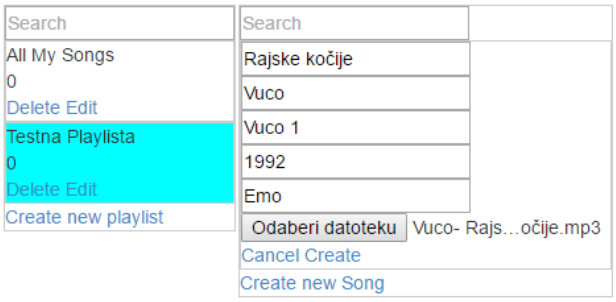
Sl. 4.5 Popis dodan ostalima

U tablici Playlist baze podataka dodan je novi popis.

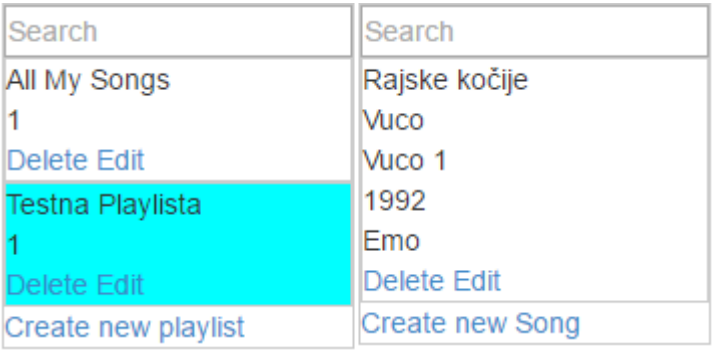
4.3Testiranje pjesama

Kod testiranja pjesama napravit će se nova pjesma u popisu pjesama Testna Playlista, nakon čega će se pjesma pojaviti u listi pjesama i u Testna Playlista i u All my Songs popisu u kojem se nalaze sve pjesme. Izgled popunjenog obrasca vidljiv je na slici 4.6. , izgled nove liste na slici

4.7. a novo stanje u tablici Song u bazi podataka na slici 4.8, dodanu pjesmu u mapi korisnika može se vidjeti na slici 4.9.



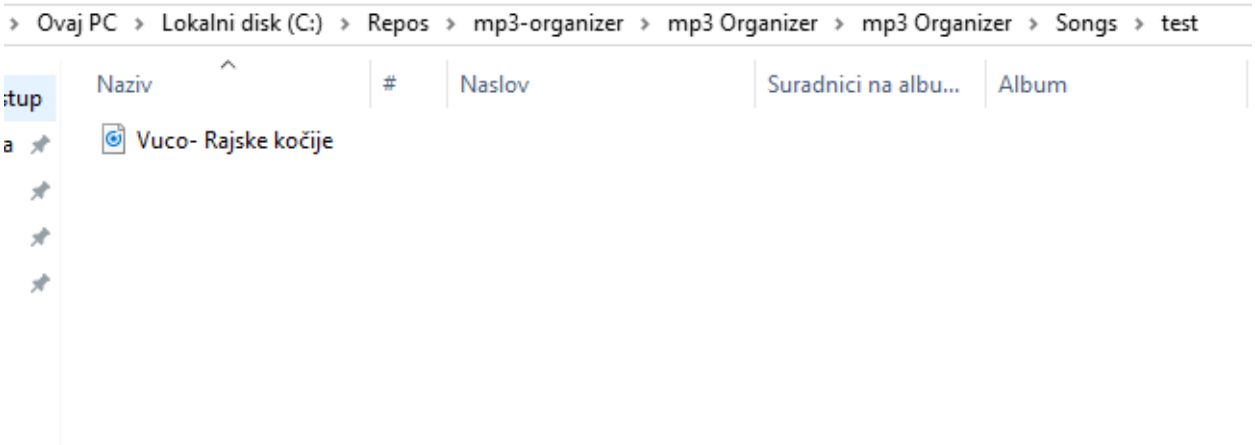
Sl. 4.6 Obrazac nove pjesme



Sl. 4.7. Nova pjesma dodana u oba popisa

SongId	song_name	artist_name	album	year	lyrics_link	korisnicko_ime	zanr	lokacija
69	jhjkdsa	jjkasdkjah	kasldjlajs	1234	NULL	new_user	jhjkashjkdhaskj	~/Songs/new_...
70	Crna ženo	Vuco	Vuco1	1992	NULL	new_user	Rock	~/Songs/new_...
71	Luciano	Pavarroti	Italia	1976	NULL	new_user	Opera	~/Songs/new_...
72	Rajske kočije	Vuco	Vuco 1	1992	NULL	test	Emo	~/Songs/test/V...

Sl. 4.8. Podaci tablice Song u bazi

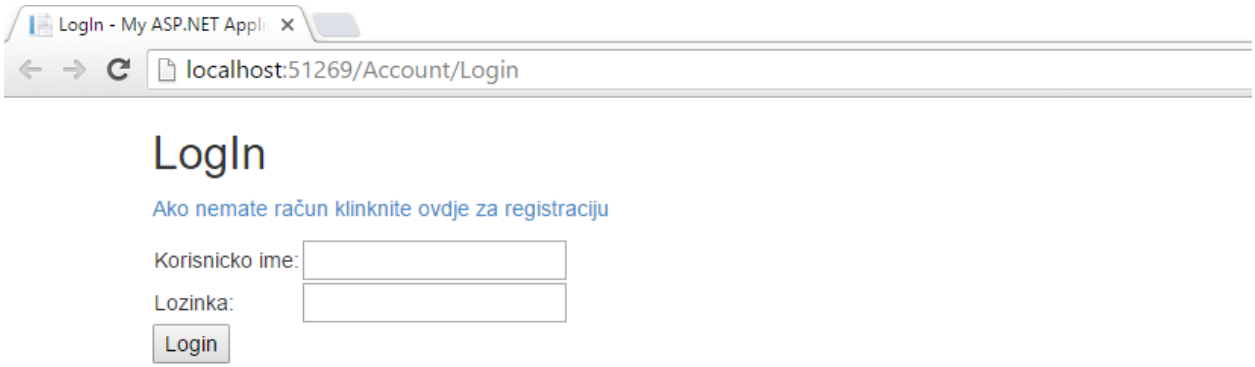


Sl. 4.8. Nova datoteka u mapi

4.4 Izgled aplikacije

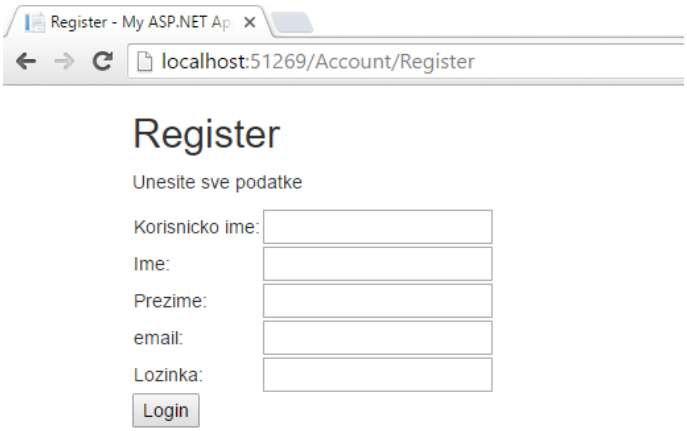
Aplikacija se sastoji od 3 glavna pogleda, jedan koji služi za obrazac registracije, jedan za obrazac prijave i jedan u kojem se nalazi *Angular.js* aplikacija koja prikazuje popise pjesama i pjesme.

Na slici 4.9 se nalazi izgled pogleda obrasca za registraciju.



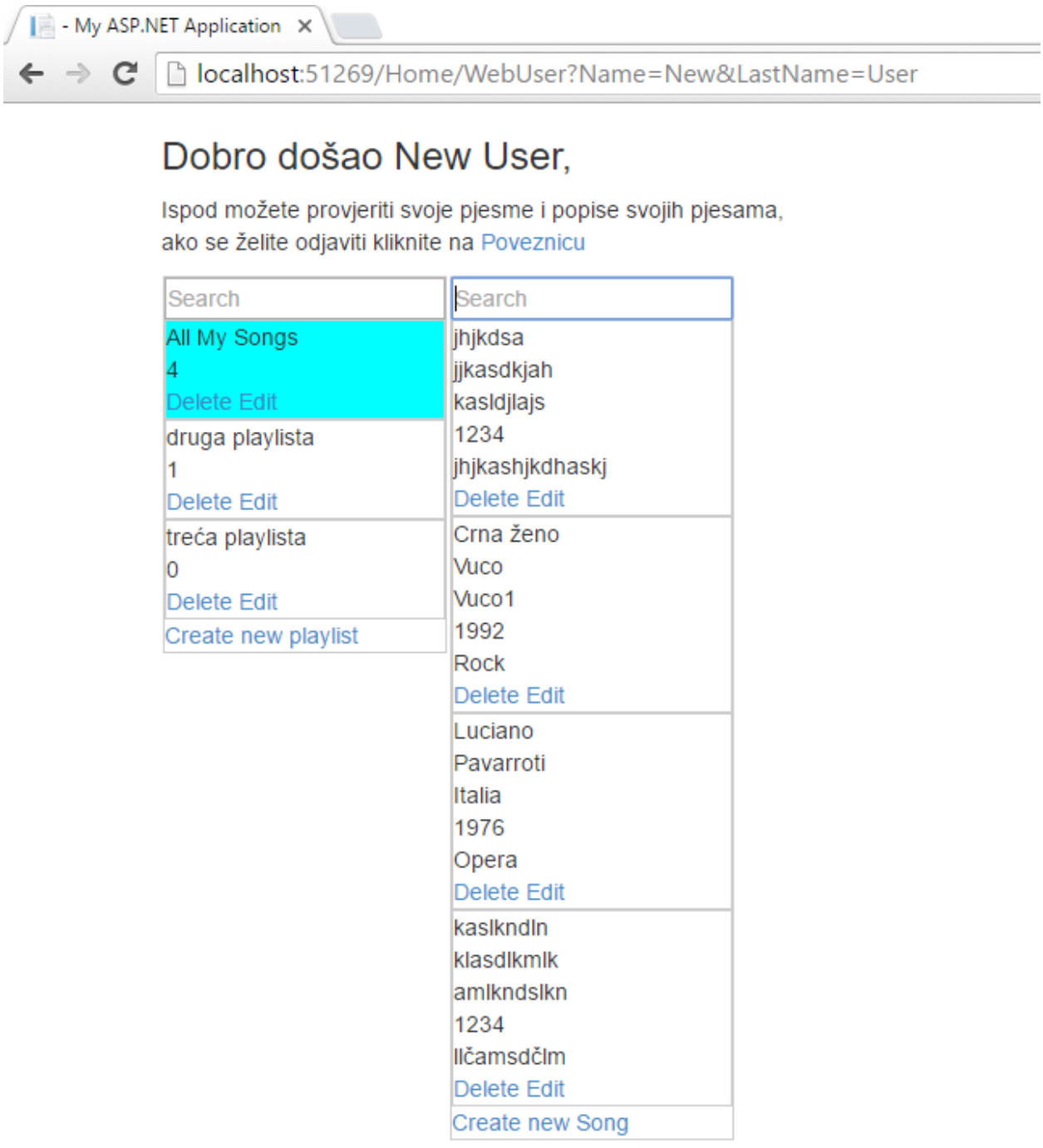
Sl. 4.9 Obrazac za prijavu

Na slici 4.10 se nalazi izgled obrasca za registraciju.



Sl. 4.10 Obrazac za registraciju

Na slici 4.11 je izgled glavnog pogleda web aplikacije, onog u kojem se nalaze popisi pjesama i pjesme u odabranom popisu.



Sl. 4.11 Pogled Angular.js aplikacije

5. ZAKLJUČAK

U web aplikaciji su ispunjeni svi zahtjevi postavljenog zadataka. Moguće je dodavati nove korisnike, svaki korisnik može dodavati svoje popise pjesama i pjesme koji su sigurni od ostalih korisnika. Dodavanje novih pjesama i popisa pjesama je jednostavno i brzo, a rad s korisnicima je osiguran na više načina pa nije moguće utjecati na tuđe pjesme i popise pjesama. Testiranja aplikacije ukazuju da aplikacija, kao i baza podataka, radi onako kako bi trebala.

Web aplikaciju može svatko koristiti a zbog toga što je vrlo brza omogućuje i dobar osjećaj tijekom korištenja. Aplikacija je ograničena zbog toga što nije moguće reproducirati pjesme, ali bi dodavanje takve opcije u web aplikaciju bilo jednostavno zbog korištenja MVC arhitekture web aplikacije. Još jedno ograničenje web aplikacije je maksimalna veličina datoteke pjesme koja ne može biti preko 10 MB.

Korištenjem MVC programske arhitekture omogućio se lagani razvoj aplikacije zbog podjele na modele, upravljače i poglede. Upravo tom podjelom se ostvaruje višeslojnost aplikacije i modularnost programskog kôda, a samim time se pospješuje i lakša nadogradnja aplikacije u budućnosti.

LITERATURA

[1] Microsoft,12.6.2016.
https://msdn.microsoft.com/en-us/library/4w3ex9c2.ASPx
[2] Neudesic, LLC, 14.6.2016.
http://www.ASP.net/mvc
[3] Ecma International, 14.6.2016.
http://www.json.org/
[4] W3schools, 15.6.2016.
http://www.w3schools.com/ASPnet/mvc_controllers.ASP
[5] Google, 15.6.2016.
https://angularjs.org/
[6] Microsoft, 15.6.2016.
https://www.microsoft.com/en-us/server-cloud/products/sql-server/
[7] Microsoft, 15.6.2016.
https://msdn.microsoft.com/en-us/library/ff400235(v=vs.100).ASPx
[8] V. Gopalakrishan, Microsoft 16.6.2016.
http://www.iis.net/learn/extensions/introduction-to-iis-express/iis-express-overview
[9] Microsoft, 17.6.2016.
https://msdn.microsoft.com/en-us/library/cc716685(v=vs.100).ASPx
[10] Microsoft, 17.6.2016.
https://msdn.microsoft.com/en-us/library/bb399249(v=vs.100).ASPx
[11] Jenny Louthan, 17.6.2016.
https://uncorkedstudios.com/blog/multipartformdata-file-upload-with-angularjs

SAŽETAK

Ovaj rad predstavlja web aplikaciju za pregledavanje i spremanje raznih pjesama i popisa pjesama. Aplikacija se zasniva na MVC programskoj arhitekturi, poslužitelj je izgrađen na ASP.NET platformi i napisan u C# jeziku, dok je web preglednik programiran Javascript jezikom uz Angular.js framework. Korisnički dio web aplikacije je u potpunosti napravljen u ASP.NET platformi jer za korisnički dio treba izravna komunikacija s poslužiteljem. Prikaz popisa pjesama i prikaz pjesama je napisan u javascriptu zbog toga što nije potrebna stalna komunikacija sa serverom nego djelomična. Baza podataka je s ASP.NET aplikacijom povezana preko ADO.NET entity data modela koji od tablica baze pravi c# klase.Postavke poslužitelja su namještene u WebConfig dokumentu na taj način da čitanje URL-ova odgovara MVC arhitekturi. Zbog korištenja MVC arhitekture lako je moguće proširiti aplikaciju s novim mogućnostima.

Ključne riječi: ASP.NET, MVC, Angular.js

ABSTRACT

Web application for MP3 songs database management

This paper present Web application for controlling database with mp3 songs. Application is based on MVC programming architeckture, server is built on ASP.NET platform and written in C# programming language, client side is programmed in javascrip's framework Angular.js. User part of Web Application is compeletely based in ASP:NET platform because it needs constant connection with server. Presentation of playlists and song lists is written in javascript because only partial connection with server is needed. Database is connected with ASP.NET web application through ADO.NET entity dana model which makes c# classes out of database tables. Server settings are configured in Web.Config file so server reads URL slike MVC architecture implies. Due to use of MVC architecture web application is easy to modify with new options.

Keywords: ASP.NET, MVC, Angular.js

ŽIVOTOPIS

Petar Petković rođen je u 20. svibnja 1994. godine u Požegi. Osnovnu školu završio je u Pleternici. Srednju školu, smjer tehničar za računalstvo, završio je u Tehničkoj školi u Požegi, nakon čega je položio državnu maturu. Petar pohađa preddiplomski studij računarstva na Fakultetu elektrotehnike računarstva i informacijskih tehnologija Sveučilišta J. J. Strossmayera u Osijeku. Petar razumije engleski jezik.

PRILOZI

CD sadrži

1. Programski kod
2. Završni rad (zavrsni.docx)